



*Technology Enhanced Learning:
Conformance - European Requirements & Testing*

Test Data Generation Tool

- User Manual -

Hans-Christian Herrmann
Institut für Wissensmedien, University Koblenz-Landau

Version 1.0 Draft 1

8 April 2005

1	INTRODUCTION	2
2	WRITING THE TEST INSTRUCTIONS	3
2.1	<TESTSET> ROOT ELEMENT	3
2.2	<TESTCASE> ELEMENT	3
2.3	<TESTLOCATION> ELEMENT	4
2.4	<TESTINSTRUCTIONS> GROUP	5
2.5	<TESTSIMPLETYPE> ELEMENT	5
2.6	<TESTFACET> ELEMENT	6
2.7	<TESTSTRUCTURE> ELEMENT	7
2.8	<TESTCONDITION> ELEMENT	8
3	DATA GENERATION PROCESS	9
3.1	COMMAND LINE PARAMETERS	9
3.2	LOGGING	9
3.3	DATAGEN OUTPUT	9
	APPENDIX - XML SCHEMA FOR TEST INSTRUCTIONS	10

1 Introduction

Testing a software system always include the task of feeding test data to the system and verifying that the data is processed correctly. The system being tested should accept all conformant data and detect non-conformant data. The conformant data should be handled correctly, that is the output should be as expected, while the non-conformant data should not affect the system stability and maybe the system should be able to recover and proceed for minor errors in the data.

To be able to verify some of the system's quality characteristics like correctness, reliability and robustness it is necessary to have the appropriate test data at hand. Often existing data is not available at the time of system testing, an even if real data is available it might not be adequate for the testing purposes. As a result of this fact, the tester in charge is often forced to write at least some test data that is specific for his testing needs. Writing this test data can be very time consuming and expensive as generally includes some amount of overhead work.

In case of XML documents, the tester might just want to test for specific element at a certain position to be handled correctly, but still has to write a complete XML document so that the element to be tested, conformant or non-conformant, is located inside an otherwise valid XML structure.

The *DataGen* tool is written for this specific case to reduce the effort of the tester and let him concentrate on the actual work of selecting the right test cases. The tester picks a specific element of the application profile, that he wants to run tests on and specifies the type of testing he want to perform, i.e. whether this element should be valid or invalid, and what type error should be included. With this information the DataGen tool will generate a set of XML documents and deliver them in form of an IMS content package¹.

Please note that DataGen only supports the tester by generating test data as specified. DataGen is not able to select the correct test data itself, like picking critical element that should be tested. This remains in the responsibility of the tester!

¹ <http://www.imsglobal.org/content/packaging/index.html>

2 Writing the Test Instructions

Before the actual process of test data generation with DataGen can be started, the tester must provide the test instructions in form of an XML file. This Test Instructions XML file must currently be written with an XML editor or text editor. The DataGen tool does not have a GUI or other sort of input possibility. It is currently provided as a command line tool and might be embedded in future into other tools to provide a better and more user friendly user interface.

2.1 <TestSet> Root Element

Description: The only allowed root element of all test instructions is <testSet>. It has several attribute that constitute the general setting for the test data generation process. The XML namespaces used are also defined here.

Multiplicity: Occurs exactly once in each XML test instructions file (1).

Attributes:

Attribute	Usage	Default Value	Description
applicationProfile	required		The application profile that is the basis for this data generation process.
constraintDefinitions	Optional		

Namespaces:

The following namespaces are required:

- **xmlns="http://iwm.uni-koblenz.de/xsd/tel_testv1p0"**
The default namespace for the test instructions (required).
- **xmlns:ti="http://iwm.uni-koblenz.de/xsd/tel_testv1p0"**
Unique namespace prefix for test instructions (required).
- **xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**
W3C XML schema instance namespace definition (required).
- **xsi:schemaLocation="http://iwm.uni-koblenz.de/xsd/tel_testv1p0.xsd tel_testv1p0.xsd"**
Defines the local instance of the XML schema file (required).
- **xmlns:xs="http://www.w3.org/2001/XMLSchema"**
W3C namespace for XML schema (required).
- **xmlns:cp="http://www.imsglobal.org/xsd/imscp_v1p1"**
Namespace of the IMS Content packaging specification.

Subelements:

- testCase

2.2 <TestCase> Element

Description: The <testCase> element contains the information required to create one XML document instance. Each correctly specified <testCase> element will result in one XML document instance. If not, the <testCase> could not be processed.

Multiplicity: Occurs zero or more time in the <testSet> element (0..*).

Attributes:

Attribute	Usage	Default Value	Description
tcID	required		A unique ID for this test case.
root	required		The root element of the application profile to be used. (Many specifications allow more than one root element. Choosing a random root element does not seem convenient for structured testing)

Subelements:

- testLocation
- testAddConstraint

2.3 <testLocation> Element

Description: The <testLocation> element defines a position in the derived base schema. This position is given in Form of an Xpath expression. The specified position is affected by the test instructions enclosed as subelement of the <testLocation>.

<testLocation> elements can be nested. This is necessary if elements are referenced in schemas and element within the reference should be addressed. Test instructions always refer to the innermost <testLocation> element.

Please note that although the data generation process is based on the application profile given in the respective <testSet> attribute, this profile is internally transformed into a derived schema using the Schema Transform Tool (developed by Apple). The further data generation process work on the derived schema.

Multiplicity: Occurs one or more time in the <testSet> element (1..*).

Attributes:

Attribute	Usage	Default Value	Description
location	required		The location of the element to be affected.
baseSchema	required		Contains the location of the used XML schema file. It is required only for the top-level <testLocation>.

Subelements:

- testLocation
- testInstruction; a element group containing:

- testSimpleType
- testStructure
- testCondition

2.4 <testInstructions> Group

Description: The <testInstructions> element is used as a placeholder for the implemented categories of test instructions. In the test instructions XML file this tag will never appear, the file will rather have one of the following element types in this position:

- testSimpleType
- testStructure
- testCondition

2.5 <testSimpleType> Element

Description: The <testSimpleType> element defines what kind of XML simple type value should be generated for the simple type element defined in the surrounding <testLocation> element. DataGen will then generate a document instance that contains at least one occurrence of this element in the given form.

The effect applied to the simple type value is defined by the attributes.

Multiplicity: Occurs once in the innermost <testLocation> element (1).

Attributes:

Attribute	Usage	Default Value	Description
testType	required	valid	Defines whether the generated simple type value should be conformant or non-conformant. Allowed values are: "valid" or "invalid".
property	optional		Defines which property of the simple type is affected. Possible values: • "primitive" ○ For testType="invalid": violate the primitive Type of the simpleType, e.g. use value "abc" for a primitive type "decimal". ○ For testType="valid": -irrelevant- • Value "facet": ○ For testType="invalid": violate the facets of the simple type, but not the primitive type. ○ For testType="valid": use a

			value that is valid but on the boundaries of the simple type's value range.
constraint	optional		If this attribute is used, it will be handled with highest priority. When conflicting with other attributes, this can result even in overriding the other attribute's settings. The value is a string consisting of an Xpath expression that evaluates to a Boolean value and constrains the values to be used. The expression is made up of the placeholder "@value", a Boolean operator (=, =>, !=, and, or) and a number or string. Examples: Constraint="(@value='male')" Constraint="(@value=>42)"

Subelements:

- testFacet

2.6 <testFacet> Element

Description: This subelement of <testSimpleType> defines the facets of the simple type that should be affected. It is only useful in combination with the surrounding <testSimpleType> element that has the attribute values

```
<testSimpleType testType="invalid" property="facet">
  <testFacet facetName="..." />
</testSimpleType>
```

More than one <testFacet> subelement is allowed for <testSimpleType>. DataGen will try to violate all given facets and not violate any other facets.

Multiplicity: Occurs zero or many time (0..*).

Attributes:

Attribute	Usage	Default Value	Description
facetName	required		Specifies the facet to be affected. Possible values are: "length", "minLength", "maxLength", "pattern", "enumeration", "whiteSpace", "maxInclusive", "maxExclusive", "minInclusive", "minExclusive", "totalDigits", "fractionDigits".

Subelements: *-none-*

2.7 <testStructure> Element

Description: The <testStructure> elements specify how to affect the structure of the document to be generated. They do not affect the values of the document instance.

The surrounding <testLocation> element must point to the parent of the complexType to be affected, while the complexType itself is defined in the attribute *subelement* of the <testStructure> element itself.

Multiplicity: Occurs one or more time in the <testSet> element (1..*).

Attributes:

Attribute	Usage	Default Value	Description
testType	required		Defines how this element should be affected. Possible values are: • “valid”: Generate a valid structure for this complexType. • “invalid”: Generate an invalid structure, details are specified in attribute “complexType”. • “specific”: Allows the tester to enter a raw XML fragment at the position of this element.
subelement	required		The element to be affected in form of an Xpath expression, relative to the parent element specified in the surrounding <testLocation>.
complexType	optional		Is evaluated only for testType=“invalid” and specifies the type of error to generate. Possible values are: • “minOccurs”: the number of occurrences of this complex type is less than allowed. • “maxOccurs”: the number of occurrences of this complex type is larger than allowed. • “composition”: the composition of the structure is violated (e.g. wrong element in “sequence”, not element of “choice”, missing element of “all”)

Subelements: *-none-*

2.8 <testCondition> Element

Description: The <testCondition> element is used to test for specific conditions, identified by the conditions ID for a specific location specified by the surrounding <testLocation> element.

The full functionality for this test instruction is not yet implemented.

Multiplicity: Occurs once in the <testLocation> element (1).

Attributes:

Attribute	Usage	Default Value	Description
conditionID	required		This attribute holds the unique ID of the targeted condition. The ID must be one that is defined in the application profile.
testType	optional	valid	Specifies whether this condition should be satisfied or not. Allowed values are: • “valid”: The premise of this condition should be true and all preceding premises of other modifications should be false. • “invalid”: The premise of this condition should be false and all preceding premises of other modifications should be false.

Subelements: *-none-*

3 Data Generation Process

3.1 Command Line Parameters

DataGen is invoked from the command line with the following parameters:

```
> java -jar DataGen.jar myTestInstructions.xml D:\Output
```

Java This is the java executable of your local Sun Java installation.

-jar This parameter is needed by the java executable to recognize the format of the DataGen.jar.

DataGen.jar This parameter specified the file name of the DataGen JAR file. You may have to set the correct file name, e.g. DataGen-snapshot.jar or DataGen-0.1.jar.

myTestInstructions.xml This parameter specifies the file name of the XML file containing the test instructions that should be processed.

D:\Output This Parameter specifies a directory that will be used by DataGen as output directory. In this directory the files and directory structure of the test package will be generated. Please verify that the DataGen process has write access to this directory.

3.2 Logging

DataGen will write all logging output to the console and additionally write it to the log file DataGen.log.

If case of any errors or unusual behaviour of DataGen, please have a look at the log messages. When reporting a bug to the developers, it might be useful to include the log file to the bug report.

Note: The log file is overwritten with every start of DataGen. So please make a copy of the log file if a specific log seems relevant for you.

3.3 DataGen Output

The output of DataGen is in form of an IMS content package. This content package contains the generated documents that can be identified by the ID in the filename, as given in the test instructions. Additionally a diagnostics document is generated with information on that file.

The exact format of the test package will be documented in the next release of this document.

Appendix - XML Schema for Test Instructions

Note: This XML schema is also available online at http://iwm.uni-koblenz.de/xsd/tel_testv1p0.xsd

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- filename=tel_testv1p0.xsd -->
<xs:schema xmlns:ti="http://iwm.uni-koblenz.de/xsd/tel_testv1p0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://iwm.uni-koblenz.de/xsd/tel_testv1p0"
  elementFormDefault="qualified"
  version="TELCERT TEST INSTRUCTIONS 1.0">
  <xs:element name="testSet" type="ti:ct.testSet">
    <xs:key name="testCase_id">
      <xs:selector xpath="ti:testCase"/>
      <xs:field xpath="@tcID"/>
    </xs:key>
    <xs:keyref name="cnd_ref" refer="ti:testCase_id">
      <xs:selector xpath="/*"/>
      <xs:field xpath="@ti:refTC"/>
    </xs:keyref>
  </xs:element>
  <xs:attribute name="create">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="auto"/>
        <xs:enumeration value="testCase"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="refTC" type="xs:token"/>
  <xs:attributeGroup name="attrGrp.structure">
    <xs:attribute ref="ti:create"/>
    <xs:attribute ref="ti:refTC"/>
  </xs:attributeGroup>
  <xs:group name="grp.testInstructions">
    <xs:choice>
      <xs:element name="testSimpleType" type="ti:ct.testSimpleType"/>
      <xs:element name="testStructure" type="ti:ct.testStructure"/>
      <xs:element name="testCondition" type="ti:ct.testCondition"/>
    </xs:choice>
  </xs:group>
  <xs:complexType name="ct.testInstruction">
    <xs:attribute name="testType" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="valid"/>
          <xs:enumeration value="invalid"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
  <xs:complexType name="ct.testSet">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="testCase" type="ti:ct.testCase"/>
    </xs:sequence>
    <xs:attribute name="applicationProfile" type="xs:anyURI"/>
  </xs:complexType>
</xs:schema>
```

```

        <xs:attribute name="constraintDefinitions">
            <xs:simpleType>
                <xs:list itemType="xs:anyURI"/>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
    <xs:complexType name="ct.testCase">
        <xs:sequence>
            <xs:element name="testLocation" type="ti:ct.testLocation.global" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="testAddConstraint" type="ti:ct.testAddConstraint"
minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="tcID" type="xs:token"/>
        <xs:attribute name="root" type="xs:QName" use="required"/>
    </xs:complexType>
    <xs:complexType name="ct.testLocation.local">
        <xs:sequence>
            <xs:element name="testLocation" type="ti:ct.testLocation.local"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:group ref="ti:grp.testInstructions" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="location" type="xs:token" use="required"/>
    </xs:complexType>
    <xs:complexType name="ct.testLocation.global">
        <xs:complexContent>
            <xs:extension base="ti:ct.testLocation.local">
                <xs:attribute name="baseSchema" type="xs:anyURI" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ct.testSimpleType">
        <xs:complexContent>
            <xs:extension base="ti:ct.testInstruction">
                <xs:sequence>
                    <xs:element name="testFacet" type="ti:ct.testFacet"
minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="constraint" type="xs:token"/>
                <xs:attribute name="property">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="facet"/>
                            <xs:enumeration value="primitive"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ct.testFacet">
        <xs:attribute name="facetName" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="length"/>
                    <xs:enumeration value="minLength"/>
                    <xs:enumeration value="maxLength"/>
                    <xs:enumeration value="pattern"/>
                    <xs:enumeration value="enumeration"/>
                    <xs:enumeration value="whiteSpace"/>
                    <xs:enumeration value="maxInclusive"/>
                    <xs:enumeration value="maxExclusive"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>

```

```

        <xs:enumeration value="minInclusive"/>
        <xs:enumeration value="minExclusive"/>
        <xs:enumeration value="totalDigits"/>
        <xs:enumeration value="fractionDigits"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
<xs:complexType name="ct.testStructure">
    <xs:complexContent>
        <xs:extension base="ti:ct.testInstruction">
            <xs:sequence>
                <xs:any namespace="##other" processContents="skip"
                    minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="subelement" type="xs:token" use="required"/>
            <xs:attribute name="complexType">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="minOccurs"/>
                        <xs:enumeration value="maxOccurs"/>
                        <xs:enumeration value="composition"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="ct.testCondition">
    <xs:complexContent>
        <xs:extension base="ti:ct.testInstruction">
            <xs:attribute name="conditionID" type="xs:NCName"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="ct.testAddConstraint">
    <xs:complexContent>
        <xs:extension base="ti:ct.testInstruction">
            <xs:attribute name="namespace" type="xs:token" use="required"/>
            <xs:attribute name="constraint" type="xs:token" use="required"/>
            <xs:attribute name="definition" type="xs:token" use="optional"/>
        </xs:extension>
        <!-- the namespace that the tested added-file is connected to -->
        <!-- the constraintID of the tested constraint -->
        <!-- the tested definition -->
    </xs:complexContent>
</xs:complexType>
</xs:schema>

```

--